Ronnie Malmberg

EBL 512 G3 Paloilmoitinjärjestelmän grafiikka

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Automaatiotekniikka

Insinöörityö

12.12.2013

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Ronnie Malmberg EBL 512 G3 Paloilmoitinjärjestelmän grafiikka 32 sivua + 1 liitettä 8.11.2013
Tutkinto	insinööri (AMK)
Koulutusohjelma	Automaatiotekniikka
Suuntautumisvaihtoehto	Automaation tietotekniikka
Ohjaaja(t)	Toimitusjohtaja Marko Hämäläinen Lehtori Timo Tuominen

Opinnäytetyön tarkoituksena oli toteuttaa graafinen helposti muokattavissa oleva käyttöliittymä Panasonic EBL 512 G3 paloilmoitinjärjestelmää varten. Työssä tutustuttiin ohjelmistotuotantoon yleisesti ja projektin toteutuksessa sovellettiin ohjelmistotuotannon ja ketterän kehityksen periaatteita.

Työ jaettiin neljään osaan, jotka olivat määrittely, suunnittelu, toteutus, testaus. Määrittelyssä päätettiin vaaditut toiminnallisuudet, suunnittelussa päätettiin sovelluksen ulkoasu, hierarkia, laitteet ja käytettävyys. Toteutus tehtiin Microsoftin Visual Studio 2010:lla ja Panasonicin EBLNet SDK komponenttikirjastolla käyttäen C#-ohjelmointikieltä. Testausta varten luotiin erillinen suunnitelma, joka käytiin läpi ja puutteista luotiin raportti.

Projektin lopputuloksena syntyi toimiva ilman erillistä ohjelmistoa muokattavissa oleva käyttöliittymäsovellus. Vaatimusmäärittelyssä asetetut tavoitteet saavutettiin.

Avainsanat	Käyttöliittymä, ohjelmointi, paloilmoitin

Author(s) Title Number of Pages Date	Ronnie Malmberg Graphical interface to Panasonic EBL fire alarm system 32 pages + 1 appendices 8 November 2013
Degree	Bachelor of Engineering
Degree Programme	Automation Engineering
Specialisation option	
Instructor(s)	Marko Hämäläinen, CEO Timo Tuominen, Senior Lecturer

The purpose of this thesis was to make an easy to modifiable graphical interface for the Panasonic EBL 512 G3 fire alarm system. The software was designed using basic principles of software engineering and agile programming methods

Work was divided into four phases which were specification, design, programming and testing. At specification requirements were set. The interface was designed, then programming was done with Microsoft's Visual Studio 2010 using C# programming language and EBLNet SDK. Finally software was tested.

The result was a fully functioning easy to modify software, where the fire alarm system can be controlled remotely. The software fulfills needs to have a graphical interface for fire alarm systems.

Keywords	Graphical interface, programming, fire alarm system

Sisällys

1 Johdanto

Paloilmoitinjärjestelmä koostuu keskuslaitteesta, paloilmaisimista, painikkeista ja sireeneistä. Tyypillisesti paloilmoitinjärjestelmiä hallitaan paloilmoitinkeskukselta. Nykyaikaisiin osoitteellisiin järjestelmiin on kuitenkin mahdollisuus liittää ohjelmisto, jolla järjestelmää voidaan tarkastella ja hallita etänä esimerkiksi kiinteistövalvomosta. Tätä ohjelmistoa kutsutaan paloilmoitinjärjestelmän grafiikaksi.

Panasonic EBL järjestelmien hallintatyökalu on web-pohjainen. Se täyttää lähes kaikki vaatimukset järjestelmän käyttämistä varten ja sillä on mahdollisuus suorittaa kaikki perustoiminnot. Ainoa ominaisuus, joka siitä puuttuu on järjestelmän tilan näyttäminen graafisesti rakennuksen pohjakuvan päällä. Nykyinen Panasonicin ohjelmisto ei siis ole paloilmoittimen graafinen käyttöliittymä vaan hallintatyökalu.

Työn tarkoituksena oli tehdä sovellus, josta voidaan hallita Panasonic EBL 512 G3 -paloilmoitinjärjestelmää graafisesti. Panasonic tarjoaa mahdollisuuden hallita paloilmoitinjärjestelmää liityntälaitteen ja valmiiden koodikirjastojen avulla.

Sovelluksen teko aloitettiin tutustumalla ohjelmistotuotantoon ja tekemällä sovellus ohjelmistotuotannon periaatteiden mukaan.

2 Tausta

Nykyisin lähes kaikissa kiinteistöihin asennetuissa taloteknisissä laitteissa on mahdollisuus käyttää graafista käyttöliittymää, josta järjestelmän tilan näkee nopeasti. Työn tilaajalta kysytään usein mahdollisuutta tehdä paloilmoitinjärjestelmään graafinen käyttöliittymä, ja joissain kohteissa se on vaadittu.

Panasonic tarjoaa EBL -paloilmoitinjärjestelmille käyttöliittymän, joka perustuu html-tekniikkaan. Tämä käyttöliittymä ei kuitenkaan mahdollista pisteiden näyttämistä rakennuksen pohjakuvan päällä. Tästä syystä työn tilaaja päätti kehittää oman graafisen käyttöliittymän Panasonic EBL -järjestelmälle.

Työn tilaaja on TUKESin hyväksymä paloilmoitinliike ja Panasonic paloilmoitinlaitteiden maahantuoja. Sen toiminta on koko Suomen laajuista, ja sen palveluksessa on tällä hetkellä 15 työntekijää. Tilaaja panostaa toimintansa kehittämiseen ja uusiin innovaatioihin, jolla se pyrkii erottumaan kilpailijoistaan.

3 Käytetyt menetelmät

3.1 Ohjelmistotuotanto

Ohjelmistotuotannolla (eng. software engineering) tarkoitetaan ohjelmistojen tekemiseen tarkoitettuja menetelmiä. Käsiteltäessä ohjelmistotuotantoa laajasti se kattaa kaiken ohjelmiston valmistukseen liittyvät tapahtumat, kuten prosessin hallinnan ja kaikki ohjelmoinnin menetelmät. Toisin sanoen ohjelmistotuotantoon kuuluu mikä tahansa toimenpide, joka tähtää ohjelman tai ohjelmiston valmistumiseen. Tarve ohjelmille tulee asiakkaalta tai ohjelman tuottaja luo markkinoille tarpeen omalle ohjelmistolleen. [1, s 11-12]

Ohjelmistotuotannossa tutkitaan myös ohjelmistojen rakenteellisia ominaisuuksia, kuten dokumentointia, version hallintaa ja jäljitettävyyttä. Myös ohjelmistojen määrittely-ylläpito-, toteutus ja suunnitteluprosesseja tutkitaan ja pyritään parantamaan. [3]

Yleisesti ohjelmistotuotantoon kuuluvat myös laatujärjestelmät.[1, s.12] Laatujärjestelmiä käytetään yrityksissä kuvaamaan niiden toimintatapoja. Laatujärjestelmien tavoitteena on dokumentoida ja ottaa käyttöön hyväksi havaitut toimintatavat yrityksissä sekä parantaa niiden toiminnan laatua.[4]

Ohjelmistokehitystä tehdään projektiluontoisesti. Ohjelmistojen kehitys ja ylläpito jatkuvat koko niiden elinkaaren. Projektin toteutus vaihtelee huomattavasti noudatettavan prosessimallin mukaan. Noudatettava prosessimalli voi olla joko todella kevyt tai myöskin hyvin raskas. Raskaita prosessimalleja käytetään yleensä silloin ohjelmistolta vaaditaan suurta luotettavuutta. Ja kevyitä prosessimalleja pienissä ja kustannustehokkaissa projekteissa. [1,s 19-25]

Ohjelmistotuotantoa pyritään käsittelemään systemaattisesti elinkaarimallin mukaan. Siinä ohjelmiston valmistus pyritään näkemään laajana, aikaan sidottuna prosessina. Ohjelmiston koodaaminen on vain yksi osa tätä prosessia.[3] Elinkaarimalleja on erilaisia, ja tässä projektissa käytetään vesiputousmallia. Vesiputousmallin [1, s 37 kuva 2.4] osia ovat:

3.1.1 Vaatimustenmäärittely

Vaatimustenmäärittelyssä kuvataan ohjelmistolle asetetut tavoitteet. Siinä määritellään, miten lopullisen ohjelmiston tulisi toimia ja miten nämä tavoitteet saavutetaan. Vaatimusmäärittely jaotellaan yleensä kahteen osaan, ei-toiminnallisiin ja toiminnallisiin vaatimuksiin. Ei toiminnallisia vaatimuksia ovat mm. laatuvaatimukset ja resurssivaatimukset. Vaatimusmäärittely tehdään työn tilaajan kanssa. [1, Luku 3], [3]. Vaatimusmäärittelyjä voidaan tehdä myös syklisesti, eli kaikkea ei määritellä kerralla vaan jaetaan koko järjestelmä pienenpiin osiin jotka, toteutetaan vaiheittain [10]

3.1.2 Suunnittelu

Ohjelmiston suunnittelu tarkoittaa asiakaslähtöisten vaatimusten muuntamista toimivaksi ratkaisuksi. Suunnittelun pääkohteina ovat ohjelmiston rakenne ja toimintaperiaate, toiminnot, algoritmit ja käyttöliittymät. Lähtökohtana on vaatimustenmäärittelyssä luotu dokumentti, jossa kerrotaan järjestelmältä vaaditut ominaisuudet. Ja päämääränä on tuottaa dokumentti, jossa kerrotaan miten ohjelmisto toteutetaan.

Suunnittelun tuloksena on malli tai esitys järjestelmästä joka tehdään. Kokonaisuus jaetaan kahteen osaan, jotka ovat yleis- ja yksityiskohtainen suunnittelu. Yleissuunnitteluun kuuluu järjestelmän kokonaisrakenne, yhteiset tietorakenteet järjestelmän jako moduuleihin ja moduulien välinen kommunikointi. Yksityiskohtaisessa suunnittelussa suunnitellaan kukin moduuli (luokka) ja sen rakenne sekä käytetyt algoritmit [11]

3.1.3 Toteutus

Toteutus käsittää varsinaisen koodauksen ohjelmiston suunnittelussa tehtyjen dokumenttien mukaan. Mikäli edelliset vaiheet on toteutettu hyvin, on varsinaiseen koodaamiseen käytetty aika vain 10—20% koko projektiin käytetystä ajasta. [3]

3.1.4 Testaus

Testaus on yksi ohjelmistokehityksen vaihe. Testauksen aikana tutkitaan ohjelmiston toimintaa oikeassa tilanteessa. Testauksessa on myös tarkoitus kartoittaa ohjelmiston koodi dokumentoituun muotoon.[1, s 205] [3]. Testauksessa ohjelmistosta yritetään löytää ohjelmistosta käytännössä kaikki ohjelmointivirheet ja varmistaa ohjelmiston toiminta oikein. Testauksessa havaitut virheet pyritään korjaamaan niin, että loppukäyttäjälle tulee virheetön ohjelmisto. Käytännössä kaikkia virheitä ei kuitenkaan pystytä löytämään. [3]

Testaus on tärkeä osa ohjelmistotuotantoa, mutta testaus ei ole tae ohjelmiston hyvästä laadusta. Laatuajattelu tulee olla mukana jo projektin alkumetreiltä.[3]

3.1.5 Käyttöönotto ja julkaisu

Julkaisu tehdään, kun ohjelmisto on saavuttanut tietyn toiminnallisen tilan, jossa lähes kaikki ohjelmiston toiminnot toimivat virheettömästi ja ohjelmisto on valmis julkaistavaksi. Useimmissa tapauksissa julkaisu ei voi tapahtua ennen kuin kaikki ohjelmistoa kehittävät tahot ovat tyytyväisiä. [3] Joskus ohjelmiston julkaiseminen ajallaan on tärkeintä, ja tällöin karsitaan aikaa muista vaiheista, kuten testauksesta. [3]

Käyttöönotossa ja julkaisussa ohjelmistosta valmistetaan asennettava kokonaisuus, joka voidaan suunnitellusti asentaa määrätylle laitteelle. Käyttöönottoon liittyy myös laitteiston ja oheislaitteiden asennus, tietoyhteyksien valmistelu ja käyttäjän valmentaminen [3]

3.1.6 Ylläpito

Ohjelmistojen ylläpito ei kuulu ohjelmistotuotantoon, vaan ylläpidon tarkoituksena on pitää ohjelmisto toimintakuntoisena ja raportoida virheistä. Ylläpito käsittää ne toimet, jolla asiakas pidetään tyytyväisenä ohjelmistotuotteeseen. Laadukkaasta ohjelmistosta löytyy ohjelmistovirheitä ja asiakkaan tarpeet saattavat muuttua. Tämän takia käyttäjät odottavat ohjelmistoista uusia versioita. [3]

3.2 Ketterä kehitys

Ennalta suunnittelun vaikeus on noussut esteeksi perinteiselle suunnittelun mallille. Iteratiivinen eli toistava prosessi antaa mahdollisuuden muuttaa projektia hallitusti. Ohjelmisto ei välttämättä tule kerralla valmiiksi, vaan ohjelmiston kehitys jatkuu julkaisun jälkeen.[5]

Periaatteena on, että jokaiseen vaiheeseen sisältyy oma pieni projekti. Näihin projekteihin sisältyvät samat vaiheet kuin jokaiseen ohjelmistotuotannon projektiin, eli vaatimusmäärittely, suunnittelu, toteutus ja validointi. Etuna on se, että pienissä projekteissa voidaan suunnata ja korjata tehtyä kehitystyötä mahdollisimman aikaisessa vaiheessa. [5]

3.3 Prototyyppimenetelmä

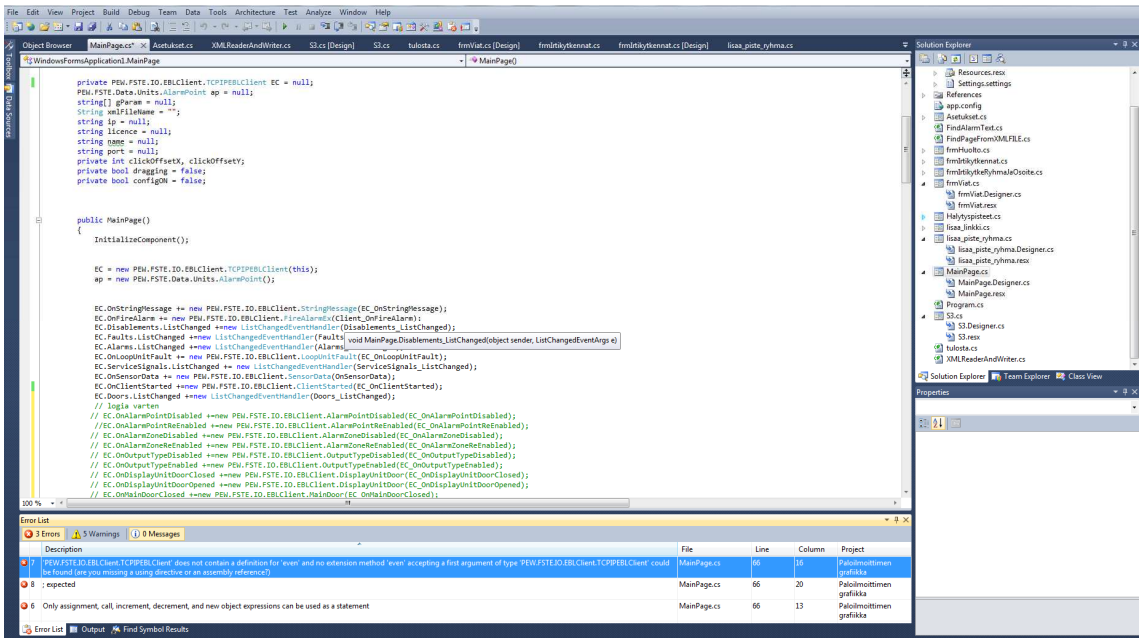
Prototyyppimenetelmässä ohjelmiston kehitys voidaan tehdä spiraalimaisesti. Eli ensin ohjelmiston käyttöliittymä, jonka jälkeen tehdään toiminnallinen osuus. Periaatteena on hyväksyttää jokainen vaihe työn tilaajalla, jolloin tilaaja voi vielä esittää muutoksia. Prosessi jaetaan kehityskierroksiin, jotka tilaaja hyväksyy. Jokaisen hyväksytyn kierroksen jälkeen laajennetaan kehitystä, kunnes päädytään ohjelmiston lopulliseen versioon. [1, luku 2.4], [3]

3.4 Visual Studio

Microsoftin Visual Studio on kehitysympäristö. Sitä käytetään komentorivi- ja graafisten ohjelmistojen kehitykseen. Visual Studiota käytetään muun muassa seuraavanlaisiin sovelluksiin, Windows Forms, Windows Presentation Foundation, komentorivi, websivut, web-aplikaatiot, web-palvelut. Ja se tukee seuraavia alustoja, Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework ja Microsoft Silverlight [7]

Visual Studio sisältää työkalut koodin muokkaamiseen, kääntämiseen ja virheen etsintään. Integroitu virheenetsintä- ja korjausohjelmisto toimii lähdekoodin tasoisena, ja sillä voi pysäyttää sovelluksen kulun valittuun kohtaan, sekä tarkistaa muuttujien ja

rekisterien arvoja (kuva 1). Muut integroidut työkalut ovat käyttöliittymän suunnittelu, web-suunnittelu, olio-suunnittelu ja tietokantatyökalut. Visual Studioon on mahdollisuus liittää lisää työkaluja liitinnäisten kautta. Visual Studio tukee monia eri ohjelmointikieliä, kuten C#, C++, VB, F#, J#.[7].



Kuva 1 Visual Studio -käyttöliittymä

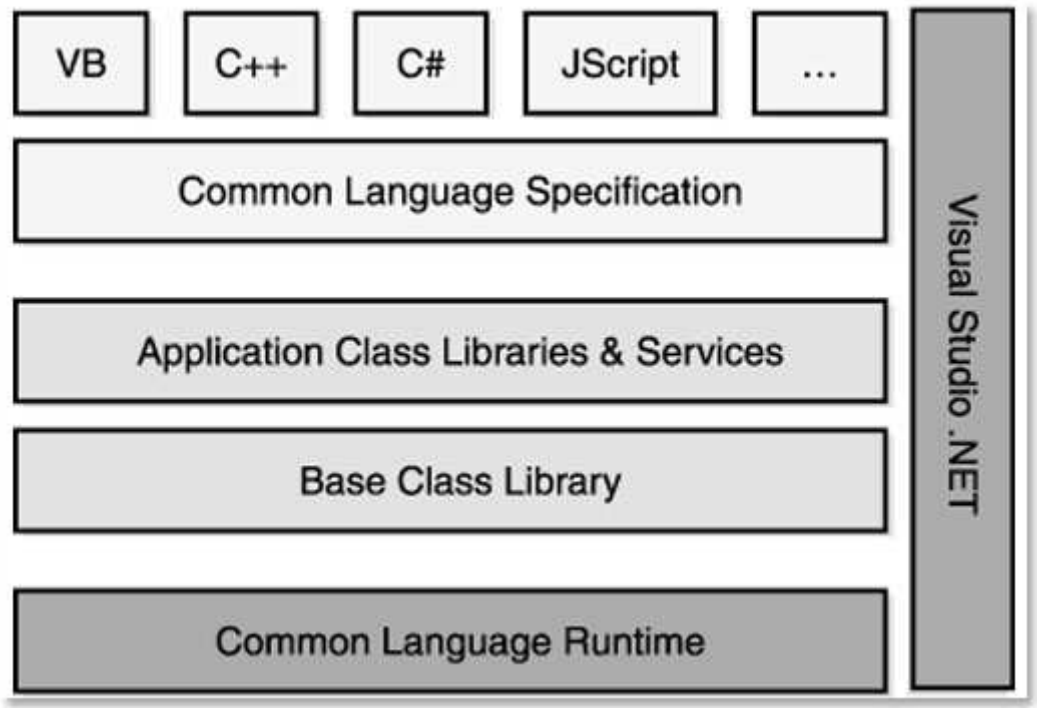
3.5 .NET Framework

.NET Framework on Microsoftin kehittämä ohjelmistokomponenttikirjasto, jota Microsoftin VisualStudio.NET -ympäristössä kehitetyt ohjelmistot käyttävät. .NET Frameworkin ideana on suorittaa mahdollisimman suuri osa ohjelmiston toiminnasta antaen ohjelmoijan keskittyä itse ohjelman tekemiseen. Joten .NET mahdollistaa ohjelmistokehityksen kohtuullisen pienellä ohjelmakoodin määrällä. [6]

.NET Framework sisältää luokkakirjastoja Windows-, Web-, Windows CE-, Windows Mobile-, Konsoli-, Service- ja Microsoft Office -tuotteisiin. Se tukee noin kahtakymmentä ohjelmointikieltä. [6]

.NET Framework määrittelee Common Language Specificationin. Tämä varmistaa saumattoman toiminnan CLS -yhteensopivien ohjelmointikielien kanssa. Tämä tarkoit-

taa, että C# -kielellä kirjoitettu ohjelma voi sisältää luokkia, jotka on kirjoitettu Visual Basic.NETillä tai Visual C++.NET:llä (kuva 2). [14]



Kuva 2 .Net rakenne

2.5 C#

C# on yksinkertainen ja moderni olio-ohjelmointi kieli. Se on hyvin samankaltainen kuin C ja C++. C# yhdistää hyvän tuottavuuden ja C++ käytettävyyden.[8]

Visual C# .NET on Microsoftin C# kehitystyökalu. Se yhdistää interaktiivisen kehitysympäristön visuaalisen käyttöliittymän suunnittelun, kääntäjän ja virheenetsintä- ja korjausohjelmiston. C# on osa Microsoftin ohjelmointituoteperhettä. Kaikki Microsoftin ohjelmointikielet tarjoavat pääsyn .NET Frameworkiin.[8]

2.6 XML

XML on rakenteellinen kuvauskieli, joka kehitettiin vuonna 1996 seuraavien asioita silmällä pitäen. Se on yksinkertainen, helposti käytettävä ja ihmisen luettavissa. Tiedos-

ton rakenteen suunnittelu on yksinkertaista ja nopeaa. Kielen oikeellisuus tulee tehdä XML mallin mukaan, joko hyvin muodostetuksi tai validiksi. Dokumentti on hyvin muodostettu, kun se täyttää vähintään seuraavat vaatimukset, sillä on yksi juurielementti, elementeillä on alku- ja loppumerkki, attribuutit ovat lainausmerkkien sisällä ja elementit eivät mene ristiin toisten elementtien kanssa. Dokumentti on validi, kun se on tehty jonkin DTD:n (Document Type Definition) mukaan. Se määrittelee sallitut muodot elementeille ja attribuuteille. Hyvin muodostettu dokumentti voi olla myös validi. [9]

XML spesifikaation liittyy seuraavat standardit (Unicode [Unicode] and ISO/IEC 10646 [ISO/IEC 10646] for characters, Internet BCP 47 [IETF BCP 47] and the Language Subtag Registry [IANA-LANGCODES] for language identification tags). Nämä tarjoavat kaikki tiedot XML-kielen ymmärtämiseen [9]

2.7 Panasonic EBLnet SDK

Panasonic EBLnet SDK on Microsoftin .NET perustuva komponenttikirjasto. PEW Nordic AB tarjoaa mahdollisuuden kommunikoida EBL 512 G3 järjestelmän kanssa. Komponenttikirjaston lisäksi kommunikointia varten tarvitaan webserver Web512G3. Koodikirjasto käyttää hyväkseen TCP/IP kommunikointia ja tarjoaa valmiit luokat paloilmoitinjärjestelmän hallitsemiseen.

Sovelluksen ohjelmointiin käytetään EBLClient luokkaa. Tämä luokka tarjoaa event handlerit ja metodit järjestelmän hallintaan lähde [12]

SDK (software development kit) on ohjelmiston kehityspaketti, joka sisältää työkalut tiettyyn järjestelmään liittymiseen tai sen kehittämiseen.[2]

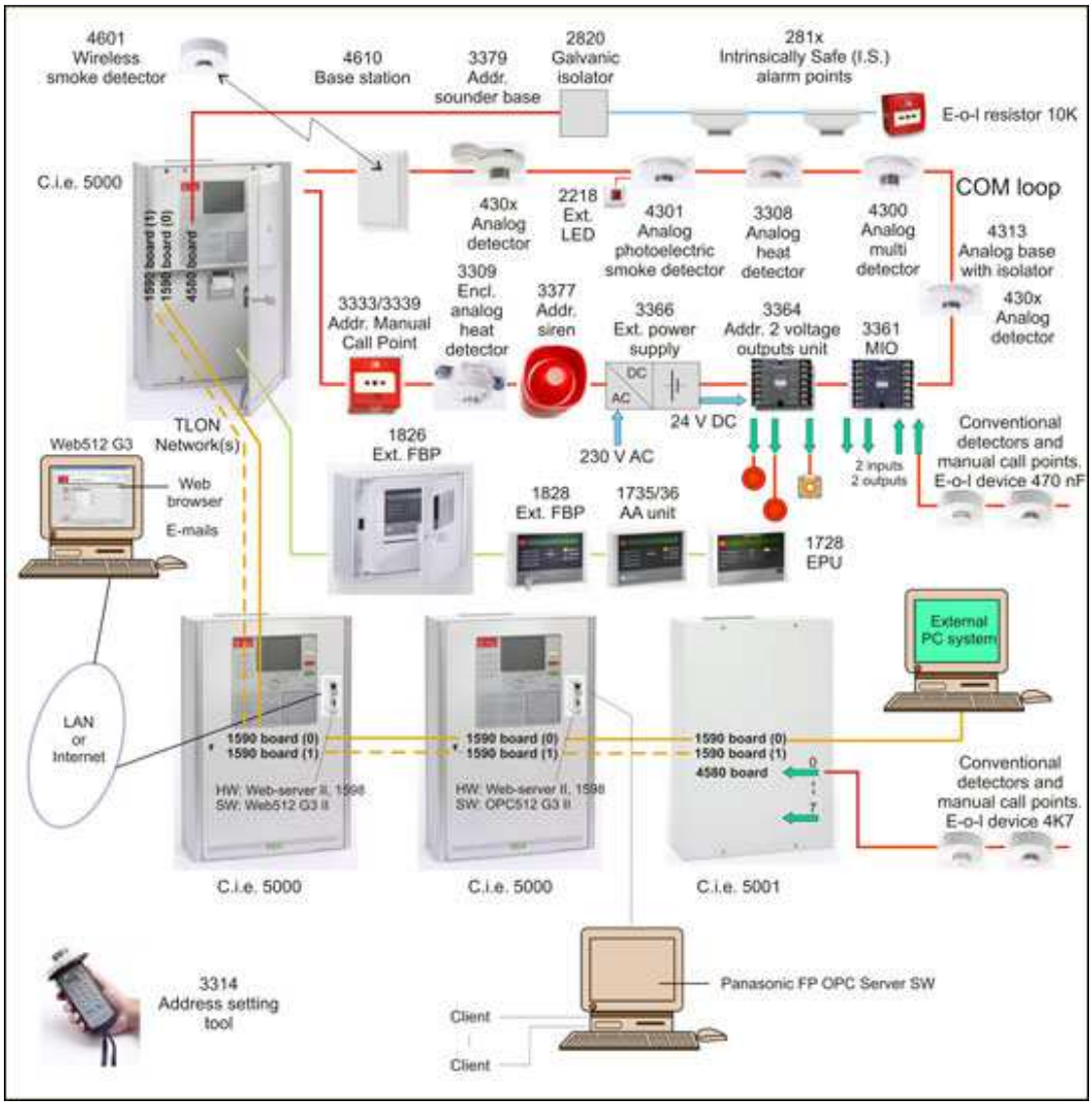
2.8 Panasonic EBL 512 G3 paloilmoitinjärjestelmä

EBL512 G3 kolmannen EBL sukupolven on analoginen osoitteellinen paloilmoitinjärjestelmä. Järjestelmä on adaptiivinen, vikavalvottu ja interaktiivinen. Järjestelmä soveltuu lähes kaikkiin kohteisiin.

Jokainen keskusyksikkö voi kommunikoida 1020 osoitteen kanssa, joista 512 voi olla hälytyspisteitä. Keskusyksikössä on neljä ilmaisinsilmukkaa, ja jokainen kenttälaite käyttää yhden osoitteen. Keskusyksiköitä voidaan liittää toisiinsa yhteensä 30 kappaletta TLONin välityksellä. Tämä mahdollistaa suuret järjestelmien koot.[13]

Jokainen järjestelmään liitetty osoitteellinen ilmaisin voidaan säätää manuaalisesti tai automaattisesti kyseiseen tilaan sopivaksi. Itsediagnostiikka tarkkailee järjestelmän toimintaa ja antaa ilmoituksen jokaisesta poikkeamasta. Järjestelmä antaa reaaliaikaista tietoa ilmaisimilta ja ilmoittaa muutoksista, palo- tai ennakkohälytyksin. Erheellisten hälytyksien estämiseksi palo- ja ennakkohälytyksiä varten on kehitetty omat algoritmit, joiden mukaan järjestelmä päättelee onko kyseessä oikea tulipalo vai erheellinen hälytys. [13]

EBLNet tarvitsee toimiakseen paloilmoitinkeskukseen liitettävän web-serverin, joka on jo itsessään web-pohjainen käyttöliittymä paloilmoittimelle. Web-server ei kuitenkaan tarjoa mahdollisuutta hälytyspisteiden näyttöön pohjakuvien päällä. Web-server tarjoaa mahdollisuuden kommunikoida ulkopuolisten järjestelmien kanssa niin, että järjestelmän hyväksynnät säilyvät (kuva 3). [12]



Kuva 3 Panasonic EBL 512 G3 järjestelmän rakenne

3 Vaatimusmäärittely

3.1.1 Sovelluksen yleiset vaatimukset

Sovellus pitää olla helposti muokattavissa ilman erillistä ohjelmistoa. Sovellukseen pitää pystyä lisätä hälytyspisteitä sekä paikantamiskaavion sivuja. Lisätyt pisteet tallennetaan erilliseen tiedostoon varmuuskopioinnin helpottamiseksi. Kommunikaatioasetukset pitää olla määritettävissä. .

Käytettävä ohjelmointikieli määräytyy EBLNet SDK mukaan. EBLNet SDK on luokkakirjasto Visual Studio -ympäristöön. Sovellus voidaan toteuttaa millä tahansa Visual Studion tukemalla ohjelmointikielellä. Esimerkki sovellukset ovat tehty C#-ohjelmointikielellä.

Paloilmoitinjärjestelmän käyttöliittymän tulee toteuttaa seuraavat vaatimukset, jotka on päätetty projektin aloituspalaverissa. Tarvittaessa vaatimuksia lisätään tai tarkennetaan.

Kun paloilmoitinjärjestelmä antaa paloilmoituksen tulee sovelluksen avata oikea paikantamiskaavion sivu, näyttää hälyttävä piste paikantamiskaaviosivun päällä niin, että hälyttävä piste vilkkuu. Hälytyspisteen pistekohtaisten tiedoista näytetään osoite, ryhmä ja pistekohtainen hälytysteksti. Paloilmoitus tulee voida kuitata ja sivun pitää olla tulostettavissa, niin että hälyttävät pisteet näkyvät tulosteessa.

Vikailmoituksessa sovellus näyttää listan kaikista järjestelmän vioista, jos vikailmoitus johtuu ilmaisimesta näytetään paikantamiskaaviosivulla vialla oleva piste. Vikailmoitukset kuitataan kaikki kerrallaan listalta.

Irtikytkennät näytetään listana, sekä kaikki irtikytketyt pisteet näkyvät paikantamiskaaviossa. Irtikytkentä tehdään joko listalta tai ryhmän / osoitteen päältä. Hälyttimet ja ohjaukset kytketään irti listasta.

Ilmaisimen huolto näytetään listana ja pisteenä paikantamiskaaviosivulla. Huoltoilmoitus kuitataan listasta.

Navigaatiosivu näyttää rakennuksen tai rakennuksien pohjakuvat kerroksittain, pohjakuvista on linkit paikantamiskaavion sivuille. Tältä sivulta on pääsy kaikkiin sovelluksen osiin, kuten asetuksiin, listoihin ja paikantamiskaavioihin.

Tapahtumaloki näyttää järjestelmässä tapahtuneet muutokset ja kirjaa muutokset erilliseen tiedostoon. Kirjaus tulee yksilöidä päivämäärän ja kellonajan mukaan.

3.1.2 Käyttöliittymän vaatimukset

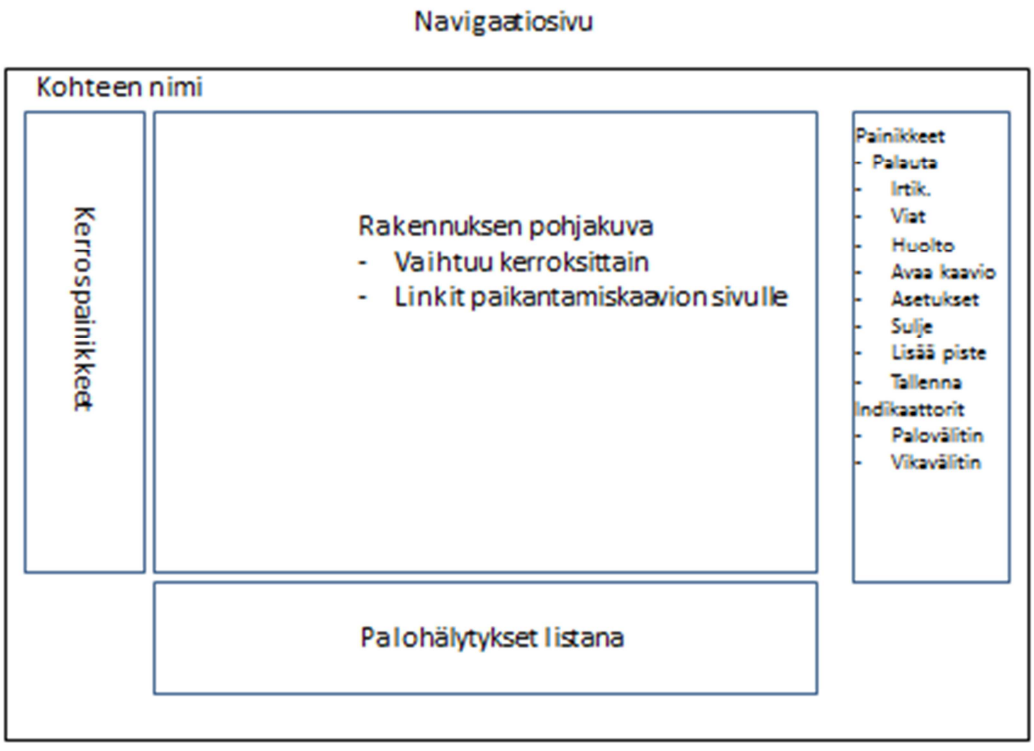
Käyttöliittymän määrittelyn pohjana käytettiin muiden toimittajien paloilmoittimen käyttöliittymiä. Sen jälkeen pohdimme yhdessä työn tilaajan kanssa, minkälainen on hyvä

käyttöliittymä, ja onko se toteuttavissa. Aloitimme jakamalla työn kolmelle eri ikkunalle, joista jokaisesta tehtiin alustava luonnos.

3.1.3 Navigaatiosivu

Navigaatiosivusta selviää koko järjestelmän tila. Ideana on jakaa järjestelmä rakennuksittain ja / tai kerroksittain. Tässä hyödynnetään jo piirrettyä järjestelmän asemakaaviota tai paikantamiskaavioita, joita vaihdetaan rakennuksittain ja kerroksittain. Jokaisen rakennuksen tai kerroksen päälle raahataan linkkejä joista oikea paikantamiskaaviosivu aukeaa.

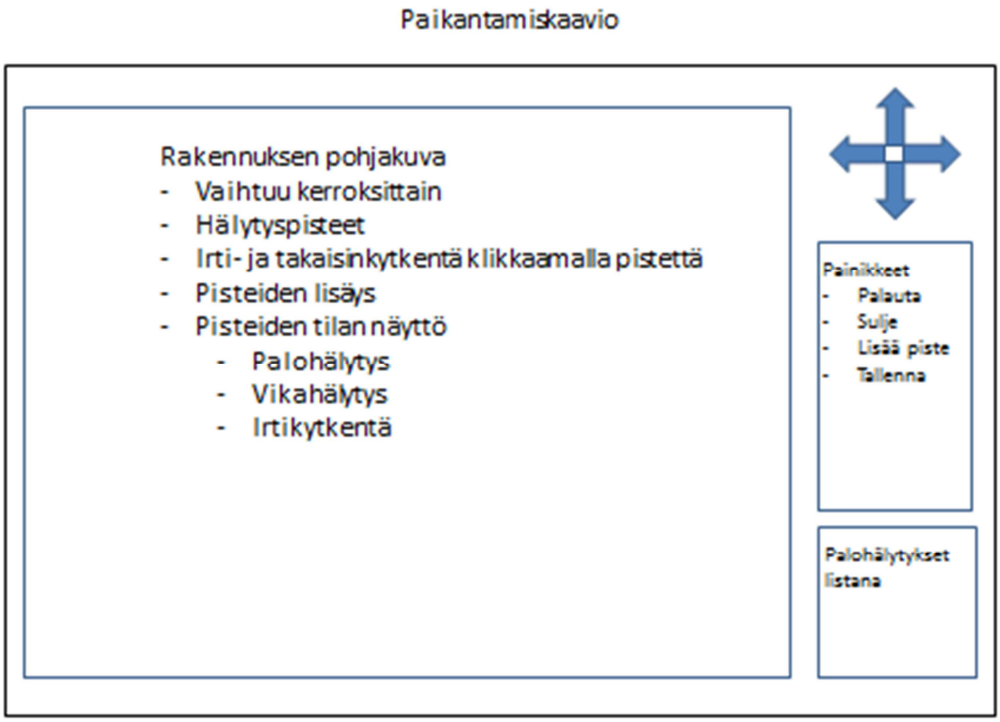
Navigaatiosivulle laitetaan painikkeet, jotka toimivat symbolien selityksenä ja mistä näkee järjestelmässä olevien vikojen, irtikytkentöjen, huoltotilassa olevien ilmaisimien määrän ja teknisten varoitusten määrän. Myös paloilmoittimen palo- ja vikavälittimen irtikytkennälle on oltava selkeät indikaattorit (Kuva 4).



Kuva 4. Navigaatiosivun alustava suunnitelma

3.1.4 Paikantamiskaavio

Paikantamiskaavio on sivu, jolla hälytyspisteitä näytetään. Valmiita paikantamiskaavioita käytetään pohjakuvina, joiden päälle hälytyspisteet raahataan. Hälytyspisteiden tilat ovat palo-, vika-, ja irtikytkentä sekä huolto. Jokaiselle tilalle valitaan oma väri tai muu merkintätapa (kuva 5). Kaikki pisteet nimetään kuten paloilmoitinjärjestelmässä eli esimerkiksi 100-01. Paikantamiskaavio sivuja on 99.



Kuva 5. Paikantamiskaaviosivun alustava suunnitelma

3.1.5 Listat

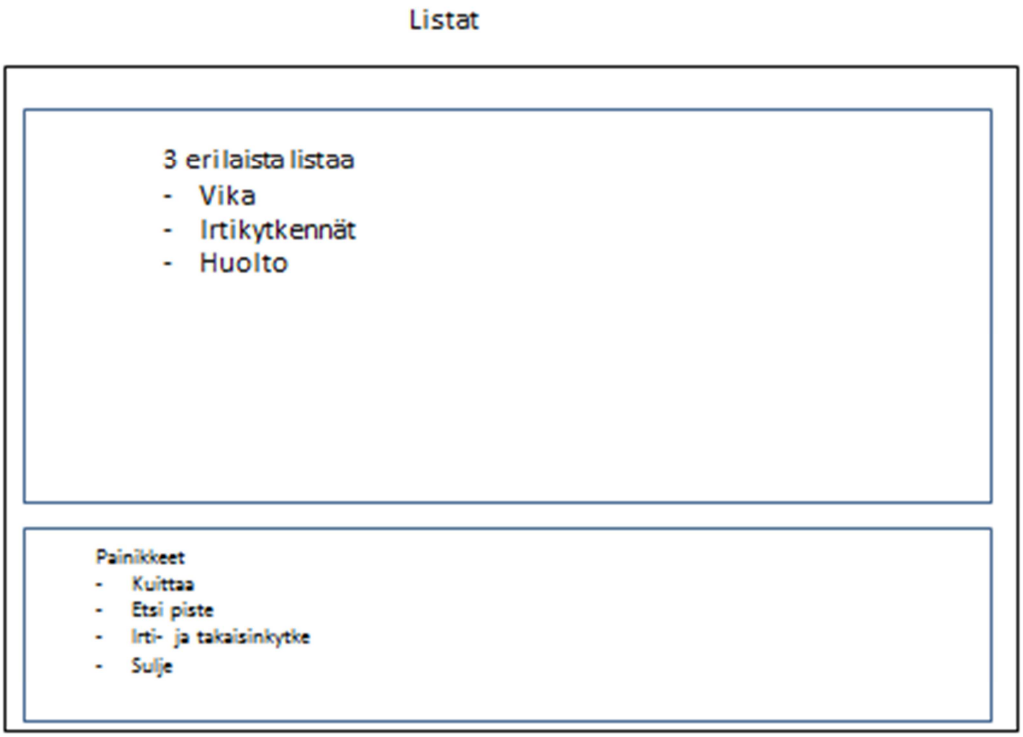
Paloilmoitinjärjestelmissä on listat seuraaville asioille, viat, irtikytkennät, ilmaisimet huoltotilassa ja tekniset varoitukset. Jolloin on loogista näyttää nämä listat myös käyttöliittymässä (kuva 6).

Vikalista sisältää tiedot vian tyypistä, ajasta ja vian tilasta. Vika voi olla aktiivinen, huollettu tai hyväksytty. Vika on aktiivinen, kun sitä ei ole hyväksytty ja vian aiheuttaja ei ole poistunut. Esimerkiksi paloilmaisin on irti kannastaan. Vika on huollettu, kun irti oleva

ilmaisin laitetaan takaisin, mutta järjestelmää ei ole kuitattu. Hyväksytty vika tarkoittaa sitä, että ilmaisin on irti ja järjestelmä on kuitattu.

Irtikytkentälista sisältää tiedot irtikytkennän ajasta ja tyypistä. Irtikytkentöjä ovat ohjauksien, hälyttimien ja hälytyspisteiden irtikytkennät. Listalla näytetään myös onko irtikytkentä tehty aikaryhmällä vai käyttäjän toimesta.

Huoltolista sisältää tiedot ilmaisimesta joka tarvitsee huoltoa ja ajan jolloin huoltopyyntö on tullut



Kuva 6. Listojen alustava suunnitelma

3.1.6 Laitteiston vaatimukset

Laitteistoksi tilaaja halusi Compulabin FITPC3 järjestelmä SSD kovalevyllä, Windows 7 pro 32 -bittisellä käyttöjärjestelmällä ja 24” näytöllä. Järjestelmän valintaa puoltaa pieni fyysinen koko ja huoltovapaus. Järjestelmässä ei ole lainkaan fyysisesti liikkuvia osia.

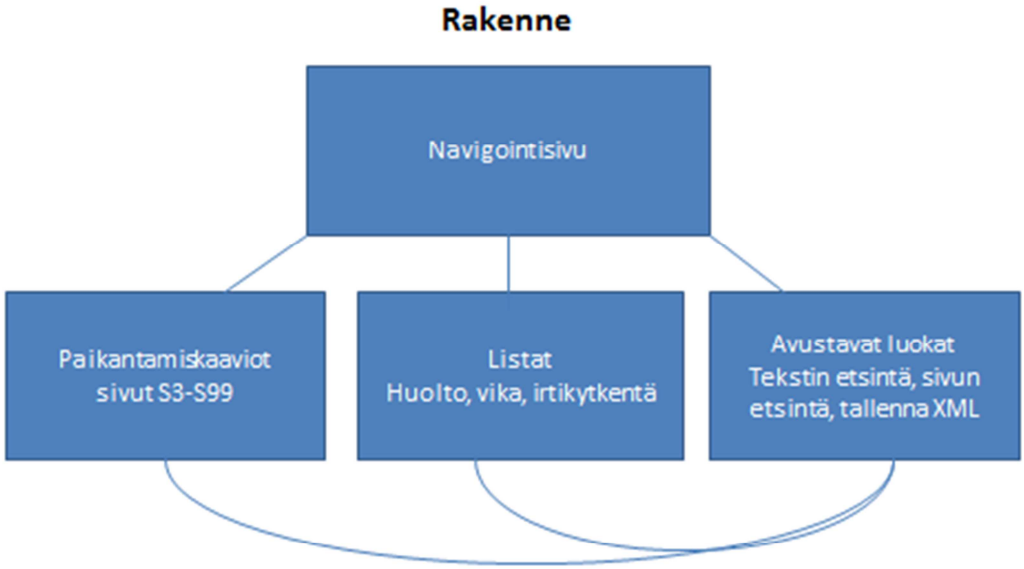
FITPC tullaan asennetaan näytön taakse vesa-kiinnikkeellä, jolloin järjestelmä ei vie kuin näytön tarvitsen tilan.

4 Sovelluksen suunnittelu

4.1 Rakenne

Sovellus suunniteltiin ajatellen, että on yksi pääluokka joka tarvittaessa avaa muita luokkia. Pääluokka on navigointisivu, joka hoitaa suurimman osan toiminnallisuudesta, kuten kommunikaation aloittamisen ja lopettamisen, hälytyksien tarkkailun, sivujen availemisen hälytyksien yhteydessä, lataa ja tallentaa XML-tiedostosta oman osuuden (kuva 7).

Paikantamiskaaviosivun luokka hoitaa pienemmän osan toiminnallisuudesta, se vahtii hälytyksiä, lisää hälytyspisteitä, lataa ja tallentaa XML tiedostosta oman osuuden, tulostaa. Muita pieniä luokkia ovat, pisteiden lisäys, pisteiden irti- ja takaisinkytkentä, hälytyspisteiden tekstin etsintä, paikantamiskaavion sivun etsintä XML tiedostosta, huolto-, vika- ja irtikytkentälista.



Kuva 7. Sovelluksen yksinkertaistettu rakenne

4.2 Kommunikointi

Kommunikointiin paloilmoitinjärjestelmän kanssa käytetään pelkästään TCPIPEblclient-luokkaa. Luokasta löytyy metodit yhteyden avaamiseen, sulkemiseen, irti- ja takaisinkytkemiseen ja palauttamiseen.

Yhteyden luominen aloitetaan asetuksien lukemisella. Kommunikaatiota varten tarvitaan IP-osoite, portti ja lisenssi. Ip-osoite ja portti saadaan Web-serverin asetuksista. Lisenssi täytyy ostaa Panasonicilta, ja se käy vain yhteen webserveriin. Yhteyden tila tulee olla valvottu. Aktiivinen kommunikointi kirjataan lokiin. Sovellus synkronoidaan kun yhteys on avattu paloilmoitinjärjestelmään.

Luokkien välinen kommunikointi pyritään toteuttamaan tapahtumakäsittelijöiden avulla ja globaaleja muuttujia pyritään välttämään.

4.3 Käyttöliittymän suunnittelu

Käyttöliittymän suunnittelu oli hyvin suoraviivaista, koska vaatimusmäärittelyssä on selkeät kuvat miltä käyttöliittymän tulisi näyttää. Käyttöliittymästä tehtiin vielä malli joka hyväksytettiin työn tilaajalla. Tässä päätettiin sovellukset taustan värit ja yleinen ulkoasu tehtiin hyvin pelkistetyksi. Työn tilaaja halusi oman logonsa näkyviin ohjelman ikoniksi.

4.4 Navigaatiosivun suunnittelu

Navigaatioikkuna on pääluokka. Tästä ikkunasta on linkit kaikille käytettäville paikantamiskaaviolle ja kaikille listoille.

Navigaatiosivu koostuu ajon aikana luoduista kontrolleista ja kovakoodatuista kontrolleista. Ajonaikana luotuja kontrolleja ovat kerroslistan objektit ja linkit paikantamiskaavioihin. Ikkunalla on kaksi tilaa, konfigurointi- ja normaalitila. Konfigurointitilassa voidaan lisätä sekä poistaa linkkejä ja muokata kerroslistaa. Muutokset tallennetaan XML-

tiedostoon, joka ladataan aina, kun ikkuna avautuu. Jos ikkunan avautuessa asetustiedostoa ei löydy, niin tyhjä tiedosto luodaan.

Navigaatiosivulla on tarkoitus esittää paljon informaatiota, jotka eivät saa kuitenkaan näkyä kokoajan. Kerroslistasta valitaan joku kerros, jolloin näytetään sen kerroksen linkit ja oikea pohjakuva. Pohjakuva valitaan kerroslistan valitun indeksin mukaan.

Linkit luodaan Label-tyyppinä ja ne nimetään paikantamiskaaviosivun mukaan. Erikoisuutena näille annetaan Tag-arvo, joka vastaa kerroslistan indeksiä. Kun kerroslistan indeksi muuttuu käydään kaikki kontrollit läpi ja laitetaan ne kontrollit joiden Tag-arvo vastaa valittua indeksiä näkyviksi, muut kontrollit piilotetaan.

4.5 Paikantamiskaaviot

Paikantamiskaavio koostuu ajon aikana luoduista kontrolleista ja kovakoodatuista kontrolleista. Kaikki hälytyspisteet ovat ajon aikana luotuja ja muut ovat kovakoodattuja. Sivulla on kaksi tilaa, konfigurointi- ja normaalitila. Konfigurointitilassa pisteiden lisääminen, siirtäminen, poistaminen ja tallentaminen on mahdollista. Paikantamiskaavion avautuessa luetaan konfigurointitiedostosta kaikki hälytyspisteet ja piirretään ne.

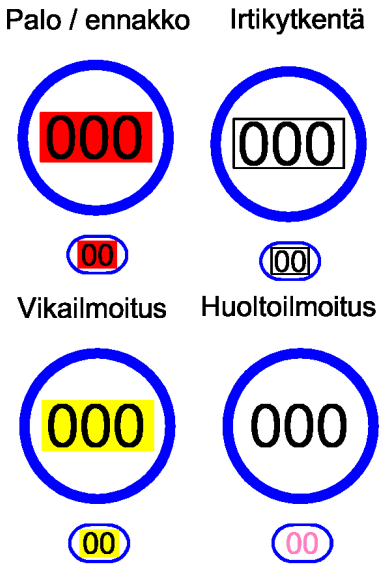
Ajonaikaiset kontrollit toteutettiin Visual Studion Label tyypillä. Ajonaikaiset kontrollit oli haastava toteuttaa niiden monipuolisuuden vuoksi. Kontrollit täytyy olla muokattavissa, eli niitä pitää pystyä lisäämään, siirtämään ja poistamaan. Kontrollit nimetään samalla tavalla kuin paloilmoitinjärjestelmä esittää hälytyspisteen, mutta varaukseksi jätettiin myös kontrollin nimeäminen samalla tavalla kuin paloilmoitinjärjestelmän tekninen numero. Jolloin voidaan tehdä myös esimerkiksi ohjauksia grafiikkaan.

Nuolipainikkeet ovat kovakoodattuja, mutta niiden linkitystä eri sivuille voidaan muuttaa konfigurointitilassa. Nuolipainikkeisiin toteutettiin muuttuvat työkaluvihjeet. Vihjeet näyttävät mille sivulle painike on linkitetty.

Paikantamiskaavion varsinainen kuva haetaan aina sivun numeron mukaan. Esimerkiksi paikantamiskaavio S3 hakee kuvansa samasta polusta josta ohjelma on käynnistetty ja kuvan nimi on oltava S3.png. Kuva skaalataan automaattisesti sopivaksi ohjelmassa. Suurikokoisten kuvien käyttö ei ole suositeltavaa niiden hitaan latauksen takia.

4.5.1 Hälytyksien esittäminen paikantamiskaaviossa

Hälytykset ilmoitetaan välkkyvällä punaisella taustavärillä, vikailmoitukset keltaisella taustavärillä, irtikytkennät symbolin ympärille piirretyllä suorakaiteella ja huoltoilmoitukset vaaleanpunaisella tekstillä (kuva 8).



Kuva 8. Hälytyksien ja ilmoituksien näyttötapa

4.6 Listat

Listat toteutetaan omina luokkina joiden rakenne on mahdollisimman yksinkertainen. Listoista pystytään kuittaamaan valitut viat, irtikytkennät, huoltoilmoitukset ja tekniset varoitukset. Listoista on mahdollisuus etsiä kyseessä oleva hälytyspiste. Listoista ei ole tallennettavaa tietoa

4.7 Asetuksien ja hälytyspisteiden tallentaminen

Tietojen tallennus tehdään XML-tiedostoon sen tarjoaman selkeyden vuoksi. Jokainen ikkuna luo oman elementin, jonka alle kirjoitetaan lapsielementteinä kontrollit. Kontrolli elementti luo itselleen lapsielementit omista tiedoistaan (kuva 9). Näin tiedostosta on helposti löydettävissä tarvittava data, kuten millä sivulla kontrolli on.

Hälytyspisteiden tekstejä ei ole saatavissa suoraan paloilmoittimelta. Tiedot päätettiin kopioida paloilmoittimen konfigurointiohjelmasta suoraan. Tekstit tallennetaan omaan tekstitiedostoon. Näistä kahdesta tiedostosta suoritetaan vertailu ja luodaan samalla oma hakemistosivu, josta voidaan varmistaa oikean hakemiston kanssa tietojen oikeellisuus.

<Grafiikka>		
	<Asetukset>	
		<kontrolli>
		xx
		xx
		</kontrolli>
		<kontrolli>
		xx
		xx
		</kontrolli>
	</Asetukset>	
	<S3>	
		<kontrolli>
		xx
		xx
		</kontrolli>
		<kontrolli>
		xx
		xx
		</kontrolli>
	</S3>	
</Grafiikka>		

Kuva 9. XML Rakenne

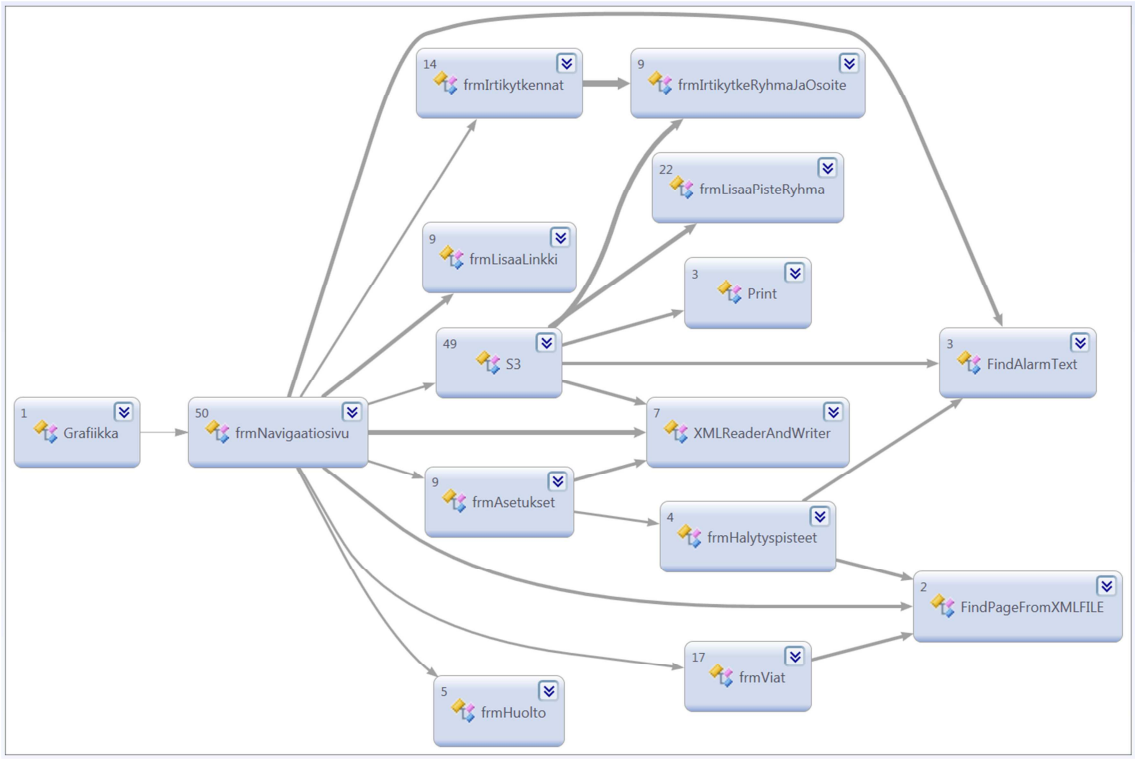
4.8 Tapahtumaloki

Tapahtumalokiin kirjataan kaikki järjestelmässä tapahtuvat asiat. Tapahtumaloki kirjoitetaan myös tekstitiedostoon. Tapahtumaloki toteutetaan listana.

5 Sovelluksen toteutus

5.1 Luokat ja hierarkia

Sovellukseen ohjelmoitiin luokat *frmNavigaatiosivu, S3-S99, frmIrtikytkennät, tfrnViat, frmHuolto*, sekä avustavat luokat *frmIrtikytkeRyhmaJaOsoite, frmLisaaPiste, frmLisaa-Linkki, frmAsetukset, frnHalytyspisteet, Print, XMLReaderAndWriter, FindAlarmText, FindPageFromXMLFILE*. Luokat on nimetty niin, että *frm* sen nimessä tarkoittaa luokalla olevan käyttöliittymä-ikkuna eli formi. Luokkien välinen hierarkia on esitetty kuvassa 10.



Kuva 10. Luokkien hierarkia

5.2 Käyttöliittymä

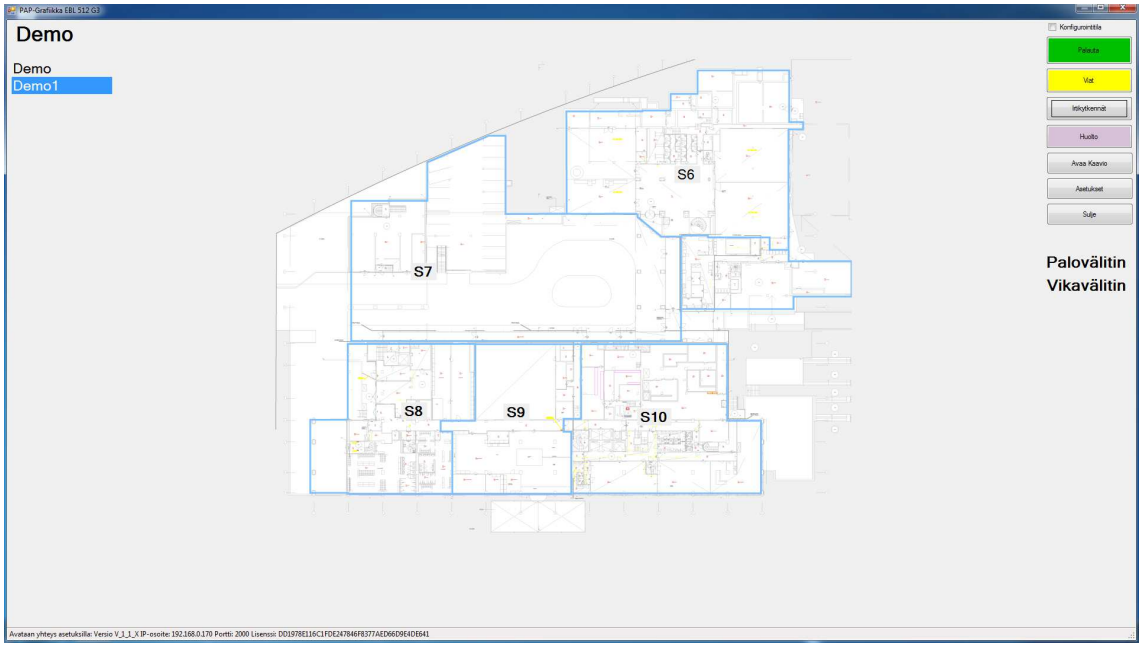
Käyttöliittymä toteutettiin käyttämällä Visual Studion peruskomponenttejä, kuten painikkeita, tekstikenttiä, pictureboxeja, listoja ja datagridviewiä.

5.2.1 Navigaatioikkuna

Navigaatiosivulle luotiin painikkeet listoja varten, picturebox pohjakuvia varten, listat kerroksia, rakennuksia ja hälytyksiä varten, tilapalkki kommunikaation tarkkailemiseksi. Dynaamisia osia ovat osassa painikkeissa olevat tekstit, kaikki listat ja picturebox (kuva 11). Luokan nimi on *frmNavigaatiosivu*.

Painikeiden teksteissä on muuttuva luku, josta näkee, montako kyseistä tapahtumaa järjestelmässä on sillä hetkellä

Konfigurointitilassa aktivoidaan painikkeita *lisää- linkki, rivi, poista rivi, tallenna* ja kenttä, jolla voidaan lisätä rivejä kerroslistaan.



Kuva 11. Navigaatiosivu

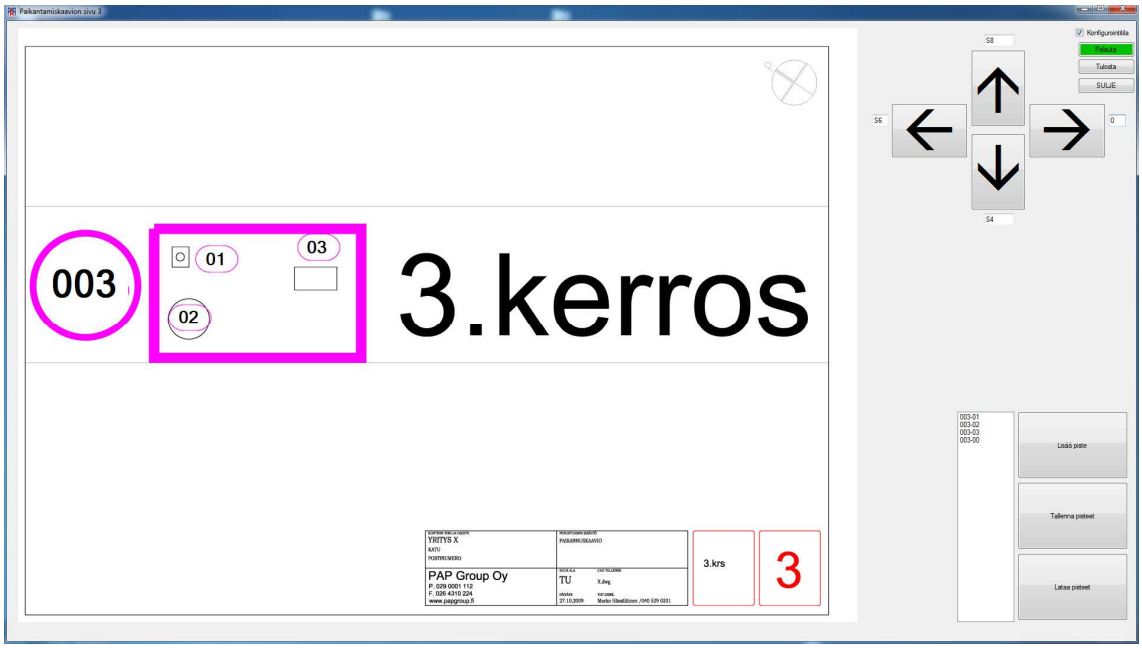
5.2.2 Paikantamiskaavio

Paikantamiskaaviosivuja luotiin 99 kappaletta jotka ovat kaikki omia luokkiaan nimeltään *S3—S99*. Jokainen sivu on perustuu samalle pohjalle ja niillä on täysin toiminnallisuudet. Sivut luotiin omiksi luokikseen, koska niiden täytyy olla helposti ladattavissa xml-tiedostosta.

Dynaamisia osia ovat picturebox, jossa on paikantamiskaaviosivun kuva png-muodossa. Kuva ladataan *paikantamiskaaviot* kansiosta luokan nimen eli paikantamiskaavion sivun järjestysnumeron mukaan. Jokainen hälytyspiste on dynaaminen ja tyypiltään label. Labelia pystyy siirtämään raahamalla. Raahaus toteutettiin käytämällä hiiren osoittimen koordinaatteja ja painikkeen painalluksia. Labelin voi tuhota klikkaamalla sitä hiiren oikealla näppäimellä.

Kovakoodattuja osia ovat kaikki painikkeet ja checkboxit. Painikkeita ovat *palauta*, joka palauttaa paloilmoituskeskuksen normaalitilaan. *Tulosta*, joka tulostaa kaaviosivun oletustulostimelle oletusasetuksilla. Ja *sulje* joka sulkee ikkunan.

Konfigurointitilassa aktivoidaan painikkeita *lisää- piste, tallenna pisteet, lataa* ja kentät jolla määritellään nuolipainikkeiden näkyvyys ja siirtyminen eri paikantamiskaaviosivulle. Myös lista kaikista sivun pisteistä aktivoidaan, jotta voidaan helposti tarkistaa kaikki sivulla olevat hälytyspisteet (kuva 12).

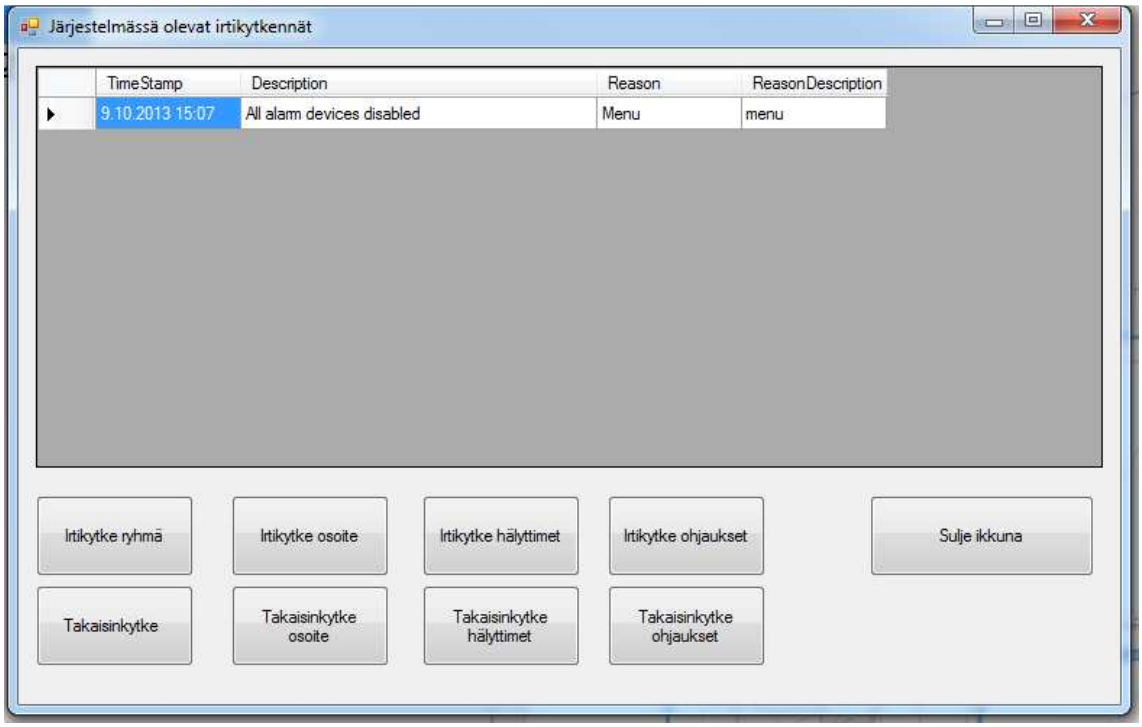


Kuva 12. Paikantamiskaaviosivu konfigurointitilassa

5.2.3 Irtikytkentälista

Irtikytkentälista tehtiin datagridviewinä, jolloin se päivittää itse itsensä ilman tapahtuman käsittelijöitä. Luokan nimi on *frmIrtikytkennät* Tästä ikkunasta voidaan suorittaa

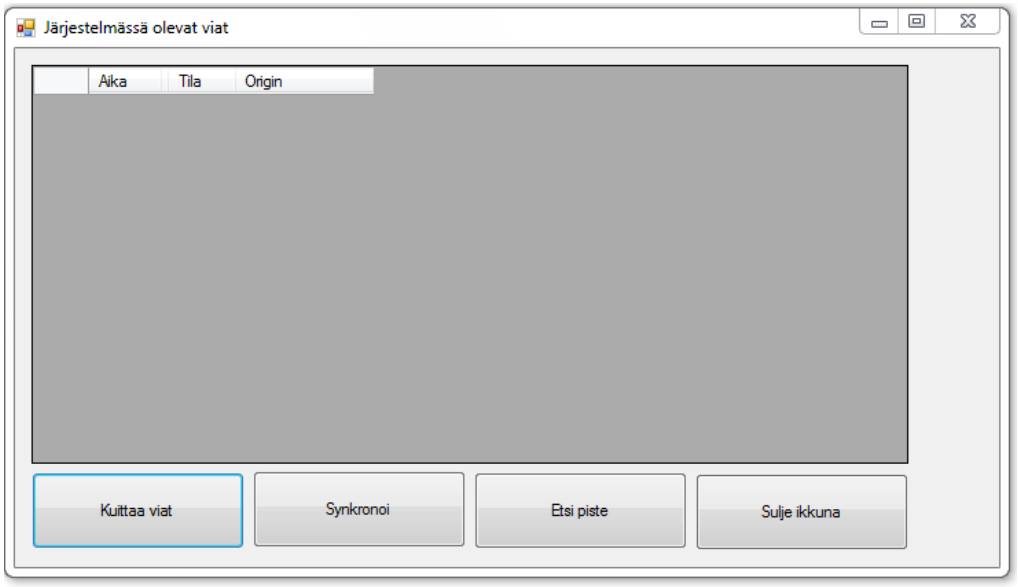
kaikki peruskäyttäjän tekemät irtikytkennät kuten, hälytyspisteen, paloryhmän, hälyttimien ja ohjauksien irtikytkennät. Lisätoimintona lisättiin etsimistoiminto, eli valitsemalla jonkun hälytyspisteen tai ryhmän irtikytkentälistasta voidaan etsi painikkeella avata kyseisen pisteen sisältävä paikantamiskaavion sivu (kuva 13).



Kuva 13. Irtikytkentälista

5.2.4 Vikalista

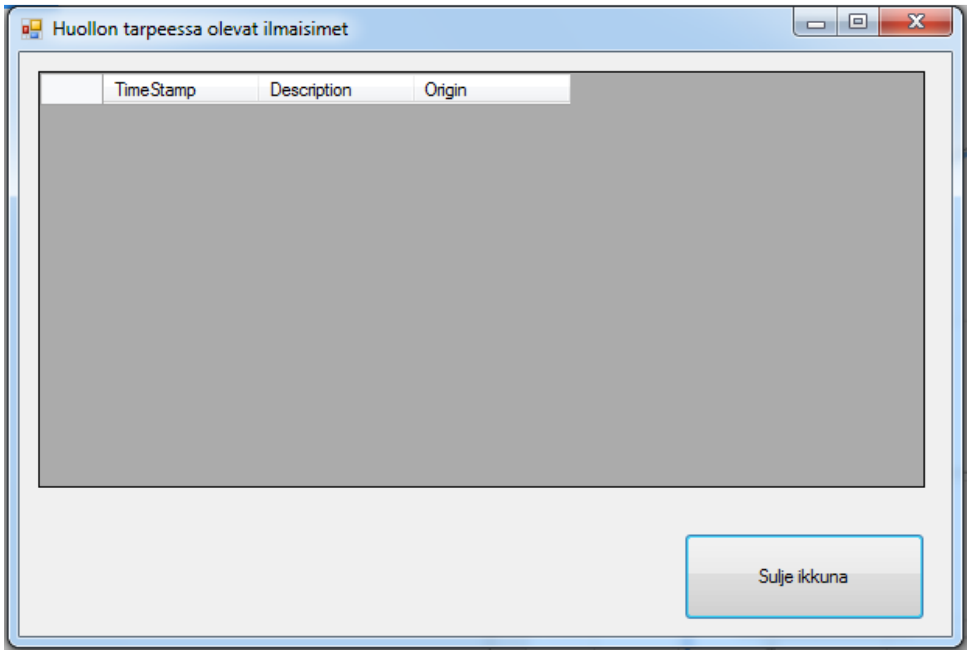
Vikalista tehtiin datagridviewinä, jolloin se päivittää itse itsensä ilman tapahtuman käsittelijöitä. Luokan nimi on *tfrnViat* Tästä ikkunasta nähdään kaikki järjestelmän viat. Vikojen kuittaus suoritetaan yhdellä painikkeella, jolloin käydään koko vikalista läpi ja kuitataan viat yksikerrallaan. Lisätoimintona lisättiin etsimistoiminto, eli valitsemalla jonkun hälytyspisteen tai ryhmän irtikytkentälistasta voidaan etsi painikkeella avata kyseisen pisteen sisältä paikantamiskaavion sivu (kuva 14).



Kuva 14. Vikalista

5.2.5 Huoltolista

Huoltolista on myös datagridview. Lista näyttää kaikki järjestelmässä olevat ilmaisimien huoltoilmoitukset (kuva 15). Luokan nimi on *frmHuolto*. Käytetyssä EBLNet SDK:ssa ei ole mahdollisuutta kuitata huoltoilmoitusta. Joten sitä ei toteutettu.



Kuva 15. Huoltolista

5.3 Avustavat luokat

Avustavien luokkien tarkoituksena on hoitaa järjestelmän pieniä toiminnallisuuksia siten, että ne ovat helposti muutettavissa lähdekoodista.

frmIrtikytkeRyhmaJaOsoite jolla irtikytketään ryhmiä tai pisteitä. Samaa luokkaa käytetään hälytyspisteen ja paloryhmän irtikytkemiseen. Hälytyspiste ja ryhmä tunnistetaan kun niitä klikataan ja luokan asetuksia muutetaan, jotta oikeanlainen näkymä avautuu.

frmLisaaPiste, jolla luodaan pisteitä paikantamiskaavioluokkiin S3-S99. Tässä luokassa on määritykset käytettävälle fontille ja lisättävän hälytyspisteen osoitteelle.

frmLisaaLinkki luokalla luodaan navigaatiosivulle linkkejä joilla avataan paikantamiskaaviosivuja. Linkit toteutettiin labelina.

frmAsetukset on luokka, jolla tallennetaan sovelluksen asetuksia, kuten ip-osoite, synkronointiaika, portti, kohteen nimi, ja EBLNet lisenssi. Ja tähän luokkaan liittyy *frnHalytyspisteet*, joka etsii kaikki hälytyspisteet paloilmoittimen ohjelmasta ja sovelluksen konfigurointitiedosta ja vertailee niitä.

Print tulostaa kuvankaappauksen oletustulostimelle tulostimen oletusasetuksilla.

XMLReaderAndWriter luo xml-tiedoston, tallentaa luokkien frmNavigaatiosivu, S3-S99 ja frmAsetukset dynaamiset kentät sekä lukee XML-tiedostoa

FindAlarmText etsii tekstitiedostosta halutun hälytyspisteen tekstin rivi kerrallaan.

FindPageFromXMLFILE etsii hälyttävän pisteen sisältävän luokan S3-S99 ja avaa sen paloilmoituksen sattuessa.

5.4 Toiminta paloilmoituksen aikana

Ohjelma avaa automaattisesti oikean paikantamiskaavion sivun config.xml tiedostossa olevan datan mukaan. FindPageFromXMLFILE luokka etsii config.xml tiedostosta labe-

lin jonka nimi vastaa palohälytyksen aiheuttanutta pistettä. FindText luokka etsii texts.txt tiedostosta hälyttävän pisteen pistekohtaisen tekstin.

Paikantamiskaavion auettua muutetaan hälyttävän pisteen taustan väriä sekunnin välein punaisesta valkoiseksi ja näytetään tekstikentässä kaikki järjestelmässä olevat palohälytykset. Hälytyksiä voi olla erilaisia, kuitenkin niiden näyttötapa on samanlainen. Hälytyksiä ovat, palohälytys, kaksois-palohälytys, ennakkovaroitus, viivästetty palohälytys jne.

Navigaatiosivulla näytetään pohjakuvat normaalisti, mutta valittaessa se pohjakuva jossa on linkki paikantamiskaavioon joka hälyttää muutetaan paikantamiskaaviosivun linkin pohjaväri punaiseksi.

6 Testaus

Testausta tehtiin paljon ohjelmoinnin yhteydessä. Näin saatiin varmistettua funktioiden toimivuus. Kuitenkin kokonaisvaltaista testausta varten tehtiin erillinen suunnitelma.

6.1 Suunnitelma

Järjestelmän testauksen suorittaa työn tilaaja. Tavoitteena on tehdä kattava testaus ja käydä kaikki järjestelmän toiminnot läpi. Testaus suoritetaan järjestelmällisesti käymällä kaikki vaadittavat toiminnot läpi aloittaen järjestelmän asennuksesta ja lopettaen sovelluksen käyttöönottoon.

Testaus aloitetaan ohjelmoimalla paloilmoitinjärjestelmä ja asentamalla sovelluksen vaatima tietokone sekä määrittelemällä Windowsin asetukset. Tämän jälkeen asennetaan sovellus ja määritellään sen asetukset vastaamaan paloilmoitinjärjestelmän asetuksia. Sovellukseen luodaan paloilmoittimen kohdekohtaisessa ohjelmassa olevat pisteet. Jonka jälkeen käydään kaikki toiminnot läpi. Läpikäytäviä toimintoja ovat irti- ja takaisinkytkentä, ennakkohälytys, palohälytys, kaksoispaloilmoitus, vika- ja huoltoilmoitus, tapahtumaloki.

Lisäksi sovelluksen kaikki painikkeet testataan ja tekstit tarkistetaan kirjoitusvirheiden varalta.

6.2 Testauksessa havaitut puutteet

Fontin koon säätäminen täytyy olla mahdollinen, koska paikantamiskaaviosivun mittakaava saattaa vaihdella kohteen koon mukaan. Labelin koko muuttuu automaattisesti siinä olevan tekstin koon mukaan. Puute korjattiin lisäämällä mahdollisuus vaihtaa fontin kokoa. Samalla konfiguraatiotiedostoon tallennetaan myös fontin ominaisuudet.

Vika- ja palovälitin eivät päivity oikein pääsivulla. Puute korjattiin ohjelmoimalla ajastin, joka tarkastaa järjestelmän tilan 30 sekunnin välein ja päivittää kentän taustavärin vastaamaan uutta tilannetta. Samalla ajastimella tarkistetaan myös onko sovelluksen ja järjestelmässä välisessä kommunikaatiossa tapahtunut muutoksia, jos muutoksia ei ole tapahtunut, yritetään yhteyden avausta uudelleen

Paikantamiskaaviosivun nuolipainikkeet jäivät näkyviin vaikka, ne eivät johdata toiselle sivulle. Puute korjattiin piilottamalla painike jos niitä ei ole määritelty linkittämään toiselle sivulle.

Vikalista ei näytä huollettuja vikoja. Tämän vian korjaamiseen kysyttiin apua EBLNet SDK kehittäjältä jolta saatiin ohjeet muuttaa IEBLClient.Faults.Latch bittimuuttuja arvoon tosi.

Ei toimi versiossa EBL 512 G3 2.0. Kehitystyön loppumetreillä Panasonic julkaisi uuden version EBLNet SDK:sta, jossa myös tuki versiolle 2.0

Pisteiden lisääminen hankalaa, joka kerralla pitää näppäillä pisteen paloryhmä ja osoite uudestaan. Pisteiden lisäysikkunaan lisättiin automatiikka joka muistaa edellisen lisätyn hälytyspisteen ja korottaa sen osoitetta yhdellä kun ikkuna avautuu.

6.3 Laitteiston valmistelu

Laitteiston valmistelu aloitettiin laittamalla FITPC kuntoon. FITPC varustettiin 64gb SSD levyllä ja siihen asennettiin Windows 7 pro 32 bittinen käyttöjärjestelmä. Järjestelmä asennettiin näytön taakse vesakiinnikkeellä, joten laitteisto on helposti siirrettävissä. Lisäksi asennettiin tulostimen ajurit ja määriteltiin verkon asetukset.

6.3.1 Windows 7 asetukset

Windowsin asetukset käytiin läpi ja asetukset määriteltiin oikeaksi. Laitteiston tulee käynnistyä ilman salasanan kyselyä suoraan sovellukseen. Kaikki virransäästötoiminnot poistettiin käytöstä ja tulostimen asetukset määriteltiin seuraavasti. Oletustulostimeen määriteltiin paperikooksi A3 ja tulostus vaakatasossa.

6.4 Sovelluksen asennus

Sovellus asennettiin luodusta asennuspaketista. Se luo itselleen tarvittavat kansiot ja tiedostot.

Sovelluksen asetukset määritellään kohdekohtaisesti klikkaamalla asetukset. Kohteen nimi, ip-osoite, lisenssi ja portti tulee antaa. Nämä löytyvät lisenssiä lukuun ottamatta webserverin konfigurointityökalusta. Webserverin konfigurointityökalusta tulee määrittää EBLNet päälle.

6.5 Kuvien tuonti sovellukseen

Paikantamiskaaviot ja navigointisivun kuvat tehdään Autocad ohjelmistolla ja tulostetaan ohjelmallisella tulostimella .png muotoon. Navigaatiosivulle kuvien nimen tulee lähteä 0:sta. Ne vastaavat navigaatiosivun listassa valittuna olevaa indeksiä. Eli kun valittuna on listan ensimmäinen kohta näytetään kuva joka löytyy hakemistosta \asema\0.png. Sovellus skaalaa kuvat automaattisesti.

Paikantamiskaaviot tuodaan vastaavasti sovellukseen ja ne nimetään sivun mukaan esimerkiksi S3.png

Kuvien koko on pidettävä mahdollisimman pienenä, jolloin kuvien latausaika on lyhyt.

6.6 Pisteiden luonti

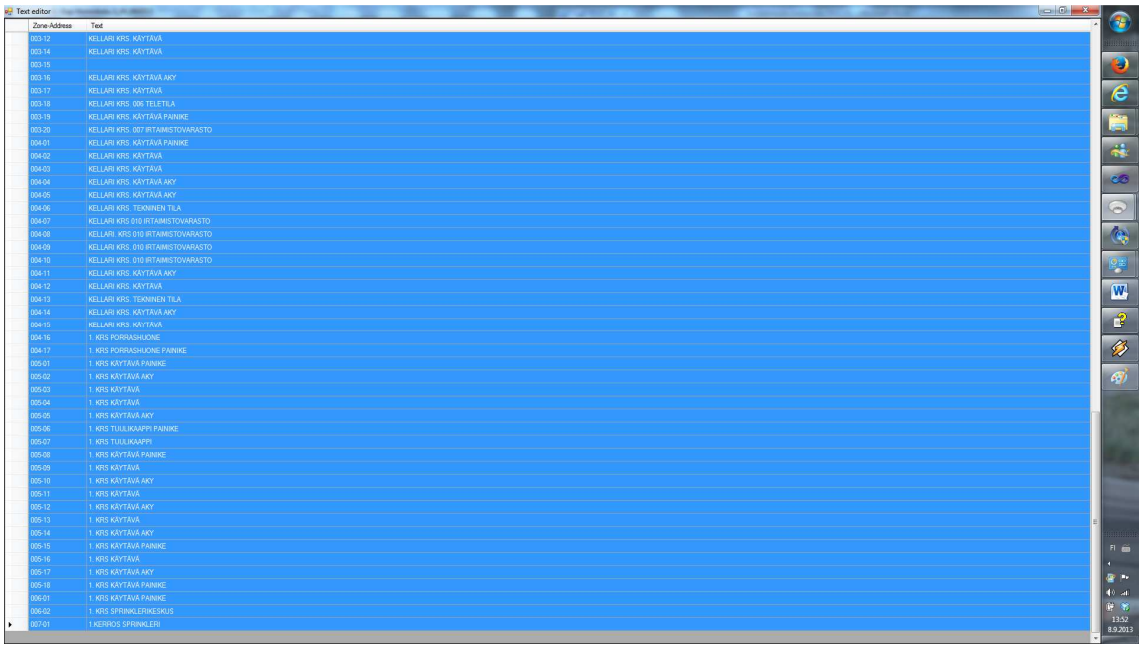
Pisteet luodaan navigaatio ja paikantamiskaavioihin seuraavasti. Valitaan konfigurointitila päälle ja klikataan lisää piste. Jolloin navigaatiosivulla aukeaa ikkuna josta lisätään linkki haluttuun sivuun ja annetaan Tag arvo. Tag määritellään samaksi kuin valittu kerros on listassa.

Paikantamiskaaviosivulla lisätään piste kuten edellä, mutta pisteitä on kahdenlaisia paloryhmä tai osoite. Lisättäessä ryhmä on tekstin koko isompi ja piste on taas pienempi. Lisättyjä pisteitä voi raahata pitämällä hiiren vasenta näppäintä pohjassa ja niitä voi poistaa klikkaamalla niiden päällä hiiren oikealla painikkeella.

Lisätyt pisteet pitää tallentaa erikseen painamalla tallenna painiketta.

6.7 Hälytystekstien tuonti ohjelmaan

Win G3 konfigurointiohjelmasta kopioidaan Alarm texts lista ja tallennetaan se tekstitiedostoon nimeltä texts.txt ja ohjelman suorituskansioon. Valinta tehdään kuvan 16 mukaan.



Kuva 16. Tekstien kopioiminen

6.8 Järjestelmän testaus oikeassa ympäristössä

Paloilmoitinjärjestelmien oman työn tarkastukseen kuuluu järjestelmän testaus, tässä testauksessa testataan järjestelmän toiminta oikeassa tilanteessa. Testaukseen kuuluu kaikkien ohjauksien, jälleenantojen ja hätäkeskusyhteyden toiminnan koestus. Ilmaisimista testataan osoitteellisissa järjestelmissä noin 30%, konventionaalisissa ryhmissä viimeinen ilmaisin ja kaikki palopainikkeet. Tästä testauksesta luodaan dokumentti nimeltään asennustodistus.

Grafiikan testaus suoritetaan oman työn tarkastuksen yhteydessä ja tarkistetaan että se näyttää kaikki pisteet oikein.

7 Yhteenveto

Projektin tarkoituksena oli tuottaa sovellus, joka toimisi paloilmoitinjärjestelmän graafisena käyttöliittymänä. Tämä tavoite saavutettiin, ja projekti tuotti vaatimusmäärittelyn mukaisen sovelluksen.

Työ jaettiin viiteen osaan, jotka olivat määrittely, suunnittelu, toteutus, testaus ja käyttöönotto.

Määrittelyvaiheessa käytiin työn tilaajan kanssa läpi kaikki ominaisuudet, mitä ohjelmassa piti olla. Työn edetessä jouduttiin myös vaatimustenmäärittelyä muuttamaan työn tilaajan halutessa uusia ominaisuuksia. Vastaavasti taas jostain ominaisuuksista luovuttiin. Alun perin tutkittiin mahdollisuutta käyttää dxf-muotoista kuvaa paikantamiskaavioiden pohjana. Tämä vaihtoehto kuitenkin jätettiin pois useammasta syystä. Dxf-muotoinen kuva perustuu vektoreihin. Kuva olisi täytynyt joka kerta luoda tyhjästä sekä sen vaikean toteuttamisen vuoksi. Lopulta päädyttiin, että pohjakuvina käytetään png-muotoisia kuvia. Ne on helppo tulostaa valmiista paikantamiskaavioista, ja niitä on helppo tulostaa myös itse sovelluksesta. Tätä vaihtoehtoa puolsi myös se, että tuloste on samalainen kuin paloilmoittimen paikantamiskaavion sivu.. Apuna määrittelyssä käytettiin työn tilaajan kokemusta erilaisista paloilmoittimen käyttöliittymistä.

Suunnittelu tehtiin prototyyppimallin mukaisesti, koska oltiin luomassa käyttöliittymää jo valmiiseen järjestelmään. Näin saatiin hyväksyttyä käyttöliittymän layout, jonka jälkeen aloitettiin toiminnallinen suunnittelu. Suunnittelu tehtiin ajatuksella, että on yksi luokka joka on koko ajan käynnissä. Tämä luokka hoitaa kommunikoinnin paloilmoitinjärjestelmän kanssa ja avaa tarvittaessa muita luokkia. Muut luokat kuten paikantamiskaavioiden sivut ja listat hoitavat toiminnallisuuden. Näitä luokkia käytettiin vain tarpeen mukaan. Näin ohjelmiston rakenne pysyi selkeänä.

Toteutusta tehtiin ketterän kehityksen ja protyyppimallia yhdistelemällä. Toisin sanoen yritys ja erehdys taktiikalla, samalla testaten koodin toimintaa. Toteutusta hankaloitti EBLNetin puutteellinen dokumentaatio, sekä EBLNetin useat eri versiot. Panasonicilta oli saatu kaksi eri versiota EBLNetistä. Jossa toinen oli betaversio missä kaikki toiminnallisuudet eivät toimineet dokumentaation mukaan. Toteutuksen haasteellisin osuus oli sovelluksen dynaamisuus ja sovelluksen tilan tallentaminen. Sovelluksen tilan tallentaminen päätettiin toteuttaa XML-tiedostona sen selkeyden vuoksi. Tähän kuitenkin harkittiin myös tekstitiedosta tehtyä ohjelmalistausta, josta kuitenkin luovuttiin sen hankaluuden takia. Tekstitiedostoon olisi ollut erittäin helppo kirjoittaa, mutta tietojen luku olisi ollut vaikeampaa, koska yksi piste vaati erittäin monta tietoa. Haastetta aiheutti pisteiden lisäys ja niiden dynaamisuus paikantamiskaavion päällä.

Testausta suoritettiin paljon ohjelmiston toteutuksen aikana. Sekä lopullinen testaus ulkoistettiin työn tilaajille. Tähän vaiheeseen tehtiin suunnitelma, jossa työn tilaajalle annettiin tehtäväksi asentaa, määritellä ja ottaa käyttöön luotu sovellus sekä koekäyttää sitä. Tästä luotiin raportti jonka, perusteella sovellusta korjattiin.

Käyttöönotto sovellukselle tehtiin, kun sovelluksesta oli korjattu edellisessä vaiheessa havaitut puutteet.

Lähteet

1 Haikala, Ilkka & Mikkonen, Tommi. 2011. Ohjelmistotuotannon käytännöt. 12., uudistettu painos. Hämeenlinna. Talentum Media

2 Auer, Antti — Auer, Liisa — Heinäsmäki, Miikka — Hölttä, Jussi — Kalliala, Eija — Laanti, Maarit — Laine, Kati — Lekman, Lare — Miinalainen, Paula — Naski, Heikki — Piiparinen, Timo — Puhakka, Hannu — Pyhäjärvi, Maaret — Pääkkönen, T. — Räisänen, Silja — Sora, Henri — Taipale, Marko — Talvio, Jukka — Tanninen, Ari — Toikkanen, Tarmo — Toivola, Towo — Toro, Kimmo — Valsta, Anne — Väyrynen, Virve — von Weissenberg,Martin. 2013 Ketterää kehitystä 1.Painos. Vantaa. Finn Lectura

3 Ohjelmistotuotanto. 2013 Wikipedia http://fi.wikipedia.org/wiki/Ohjelmistotuotanto Luettu 29.9.2013

4 Laatujärjestelmä. 2013 Qualitor http://www.qualitor.fi/palvelut/laatujarjestelma. Luettu 29.9.2013

5 Ketterä ohjelmistokehitys. 2013 Wikipedia http://fi.wikipedia.org/wiki/Ketter%C3%A4_ohjelmistokehitys. Luettu 29.9.2013

6 Microsoft .NET Framework http://msdn.microsoft.com/en-us/library/zw4w595w%28v=vs.100%29.aspx Luettu 1.10.2013

7 Microsoft Visual Studio 2010 http://msdn.microsoft.com/en-us/library/fx6bk1f4%28v=vs.100%29.aspx Luettu 10.10.2013

8 Microsoft Visual C# http://msdn.microsoft.com/en-us/library/kx37x362%28v=vs.100%29.aspx Luettu 11.10.2013

9 XML http://www.w3.org/TR/xml/ Luettu 15.10.2013

10 http://www.cc.puv.fi/~tka/kurssit/Ohjelmiston_maarittely/luennot.htm Luettu 15.10.2013

11 http://www.cs.helsinki.fi/u/laine/ot/s98/suunnittelu1.pdf Luettu 15.10.2013

12 EBLNet SDK Dokumentaatio

13 Panasonic EBL järjestelmät http://www.panasonic-fire-security.com/pewfste/en/html/1967.php Luettu 29.9.2013

14 Microsoft .NET Framework http://www.tipsntracks.com/images/tntImg/.NET-Framework-block-diagram_chap4.jpg Luettu 29.9.2013

Lähdekoodi

Lähdekoodi on salainen.